

# Práctica del Tema 5: Publicación y difusión

Blanca María Pérez Soriano

15 de julio de 2024

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Metodología empleada</b>	<b>3</b>
2.1. Software . . . . .	3
2.2. Modelo de puntos . . . . .	3
2.3. Importación a MeshLab . . . . .	3
<b>3. Resolución de la práctica</b>	<b>4</b>
3.1. Shading . . . . .	4
3.2. Triangulación . . . . .	6
3.2.1. Reconstrucción de la superficie mediante Screened Poisson	6
3.2.2. Corrección topológica de la malla . . . . .	11
3.3. Simplificación . . . . .	16
3.3.1. Simplificación: Quadratic Edge Collapse Decimation . . .	16
3.3.2. Corrección topológica de la nueva malla . . . . .	17
3.4. Parametrización . . . . .	19
3.5. Texturización del modelo simplificado . . . . .	20
3.6. Publicación del modelo . . . . .	23
3.6.1. Exportar modelo a formato X3D . . . . .	23
3.6.2. Subir nuestro modelo a GitHub . . . . .	24
3.6.3. Publicar nuestro modelo desde GitHub . . . . .	25
<b>4. Valoración personal</b>	<b>26</b>

# 1. Introducción

Creación y publicación de un modelo 3D a partir de un modelo digitalizado representado como una nube de puntos

## 2. Metodología empleada

### 2.1. Software

Para realizar esta práctica utilizaremos el programa [MeshLab](#) en la versión 64bit v2023.12.

### 2.2. Modelo de puntos

Para la realización de esta práctica se utilizará el modelo de nube puntos proporcionado en el recurso “*Modelos para la práctica*”: **MayanSculpture.ply**.

### 2.3. Importación a MeshLab

Para importar el modelo de nube de puntos seleccionado seguiremos las siguientes acciones: **File → Import Mesh → MayanSculpture.ply**

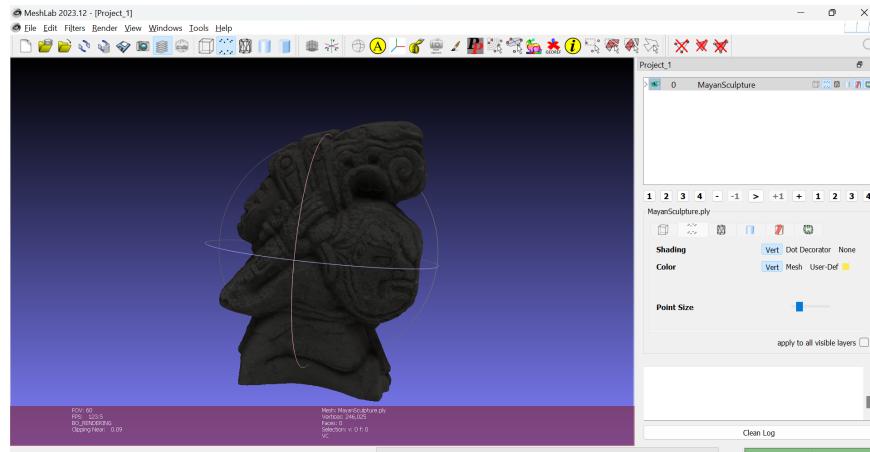


Figura 1: Malla de 246.025 vértices, 0 caras

### 3. Resolución de la práctica

#### 3.1. Shading

Para empezar correctamente, lo primero que haremos será intentar mejorar el *shading* de la pieza. Para ello calcularemos las normales, esto hará que los puntos de luz sean tomados perpendicularmente. Deberemos hacer click en la siguiente secuencia: **Filters → Point Set → compute normals for point sets**. Tras esto, se nos desplegará la siguiente ventana, deberemos hacer click en “*Apply*”:

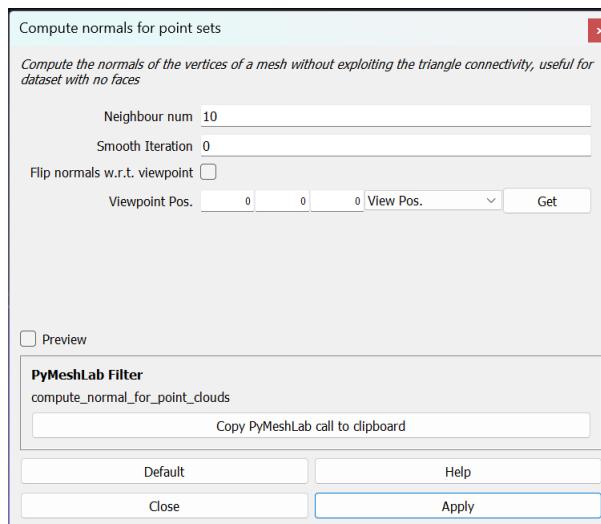


Figura 2: Panel: Compute normals for point sets

Ahora podemos ver el claro cambio de color y trazado en la pieza:

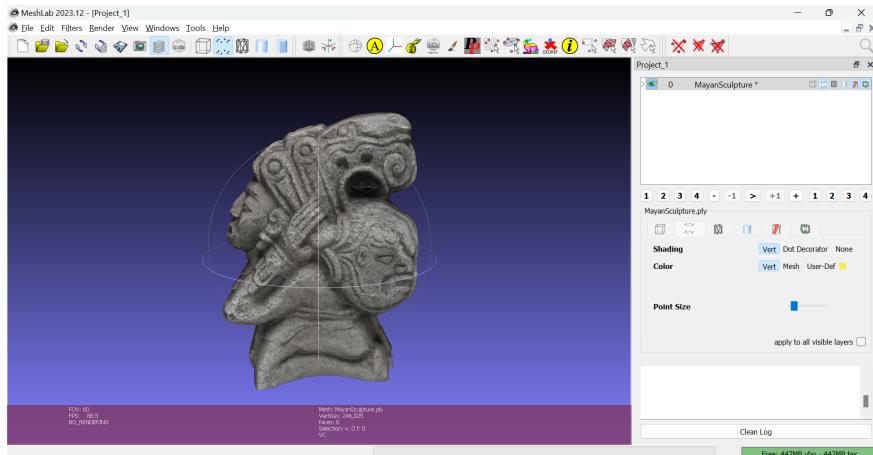


Figura 3: Modelo con normales halladas

## 3.2. Triangulación

### 3.2.1. Reconstrucción de la superficie mediante Screened Poisson

Para ello hacemos click en: “**Filters → Remeshing, Simplification and Reconstruction → Surface Reconstruction: Screened Poisson**”. Se nos desplegará la siguiente ventana:

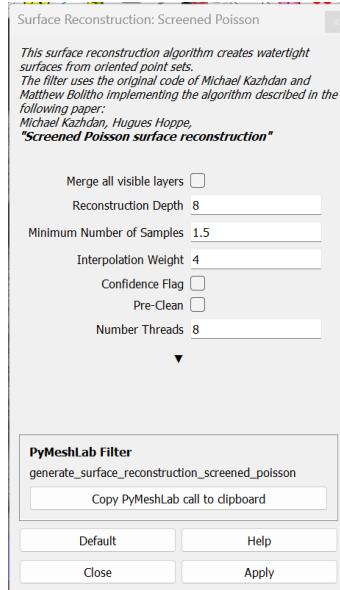


Figura 4: Panel: Surface Reconstruction

De esta ventana, los dos valores interesantes son:

- **Reconstruction Depth:** este valor lo retocaremos en base a la pérdida o ganancia de puntos tras la reconstrucción.
- **Pre-Clean:** Este *checkbox* haría un prelimiado de normales con valores nulos, en caso de ser marcado.

Como hemos visto en el curso, un buen valor de **Reconstruction Depth** para lanzar una primera reconstrucción sería el **8**. De manera que lanzamos la reconstrucción sin tocar nada más y vemos si tenemos que reajustar:

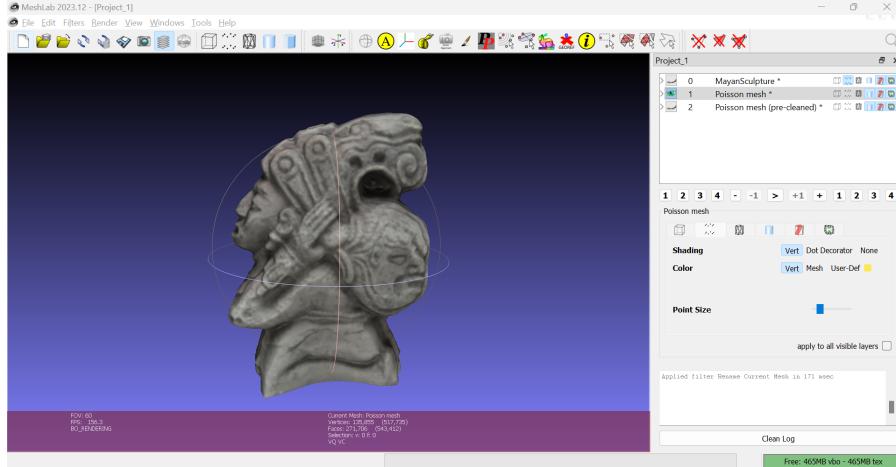


Figura 5: Reconstrucción sin pre-clean

- **vértices:** 135.855
- **caras:** 271.706

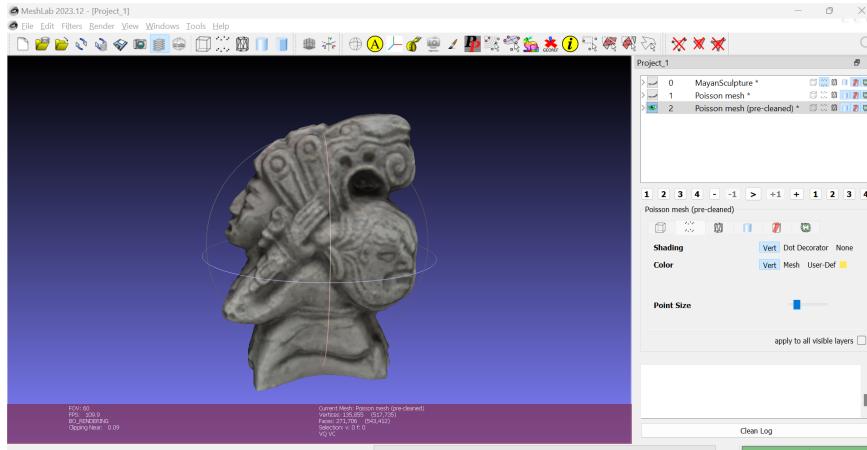


Figura 6: Reconstrucción con pre-clean

- **vértices:** 135.855
- **caras:** 271.706

La malla original tenía 246.025 puntos, de manera que hemos reducido casi en la mitad la calidad del modelo. Podemos observar, si nos acercamos a la pieza, este tipo de pérdidas:

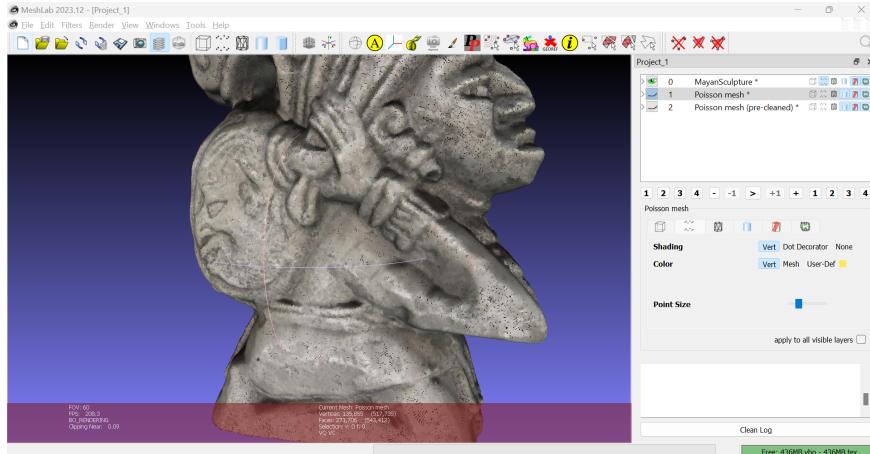


Figura 7: Por puntos

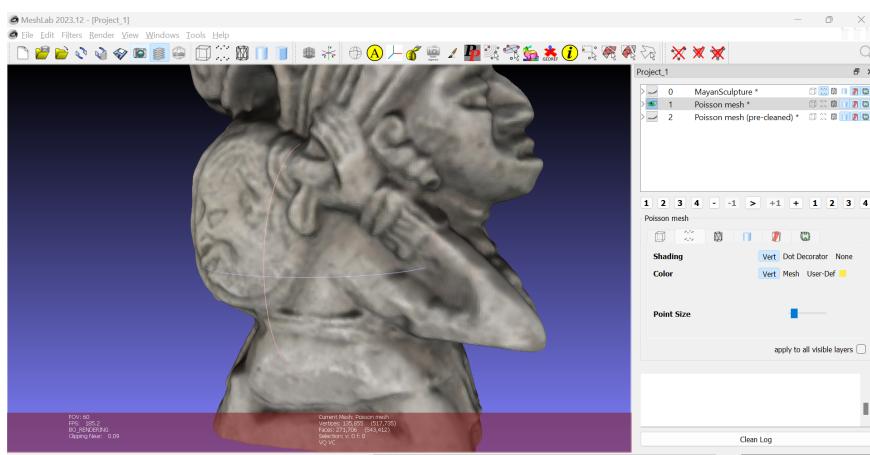


Figura 8: Poisson Mesh

Repetimos el proceso aumentando el nivel de profundidad en 1. El resultado es el siguiente:

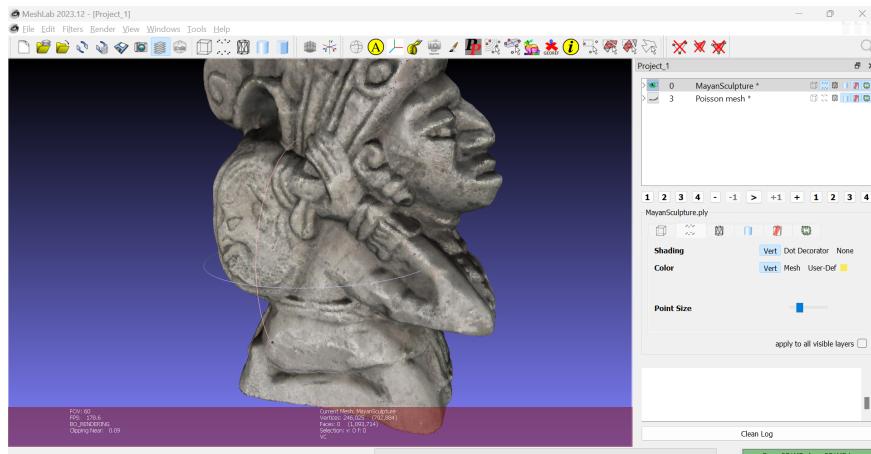


Figura 9: Por puntos

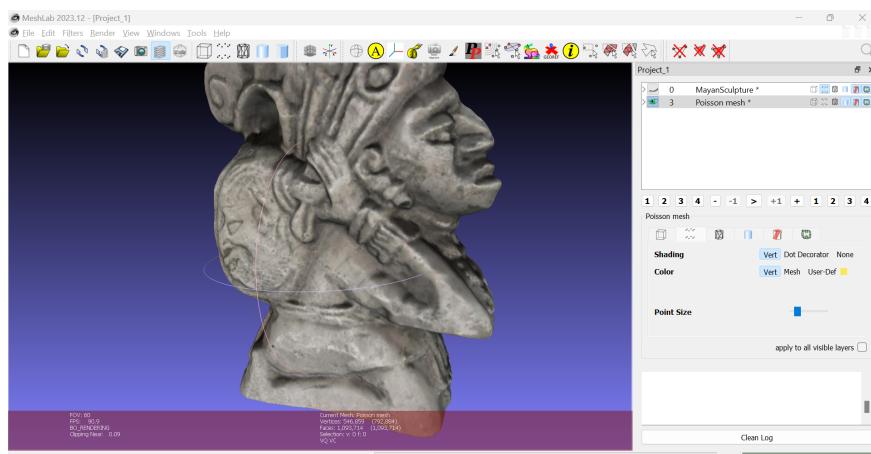


Figura 10: Poisson Mesh

Esta vez se han generado 546.859 vértices. A partir de aquí podríamos cambiar el color de la pieza, pero la verdad es que me gusta cómo está quedando.

Si hubiera querido cambiarlo, me habría movido entre estas dos pestañas del panel derecho, siendo la primera señalada la de los vértices y la segunda señalada la de las caras:

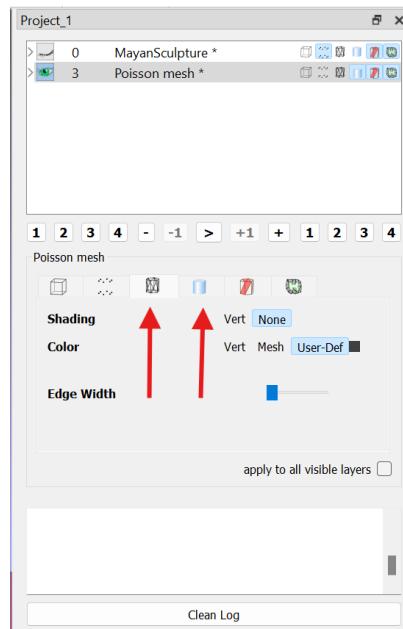


Figura 11: Pestañas modificación de color

Finalmente queda eliminar algún triángulo artificial. Para ello hacemos click en **Filters** → **Selection** → **Select faces with edges larger than...**, y el valor promedio para esta pieza es de 0.795821:

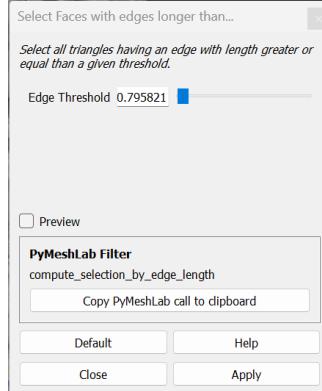


Figura 12: Panel: Eliminación de aristas largas

Tras darle a “*Apply*” veo que sólo 6 caras son seleccionadas, por lo que no tocaremos nada más de momento:



Figura 13: Poisson Mesh

### 3.2.2. Corrección topológica de la malla

Habiendo guardado una copia de la malla triangulada generada y habiendo eliminado la malla de nube de puntos, duplico la malla triangulada dentro del programa para prevenirnos del efecto de malas acciones, y empezamos con las técnicas.

### 3.2.2.1 Autointersecciones

Para detectar **autointersecciones** hacemos clicks en los siguientes apartados: **Filters → Selection → Select self intersecting faces**. Tras este proceso, veo que no se han seleccionado caras por lo que no aplico ninguna eliminación:

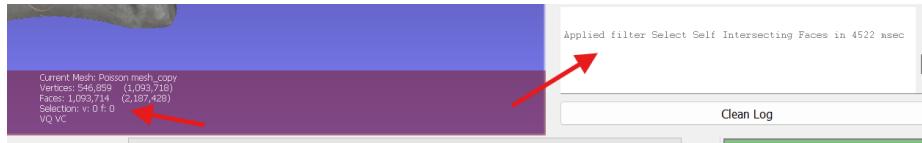


Figura 14: Panel: Autointersecciones

### 3.2.2.2 Elementos no-variedad



Figura 15: Aristas no-variedad



Figura 16: Vértices no-variedad

No sale nada para ninguno de los dos, así que parece todo perfecto por ahora.

### 3.2.2.3 T-Vértices

Para limpiar los vértices que conectan con el punto interior de una arista haremos click en **Filters → Cleaning & Repairing → Remove T-Vertices**, nos saldrá un panel y deberemos hacer click en “*Apply*”. Para este caso sí nos han salido algunas ocurrencias:

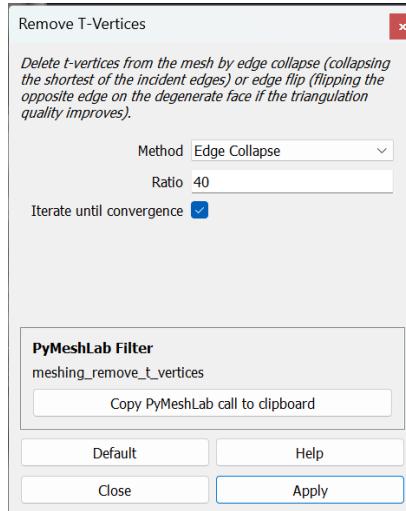


Figura 17: Panel de T-Vértices

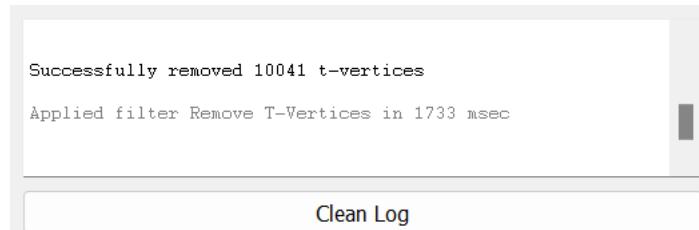


Figura 18: Log

### 3.2.2.4 Filtrados varios

En este apartado dejaré fotos de algunos filtrados más que se han hecho antes del tapado de fisuras:

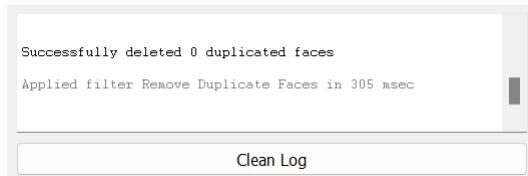


Figura 19: Remove duplicated faces

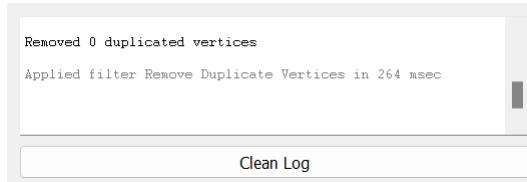


Figura 20: Remove duplicated Vertices



Figura 21: Remove isolated pieces

### 3.2.2.5 Tapado de fisuras

Una vez tenemos la malla limpia, procedemos a tapar posibles agujeros. Para ello vamos a **Filters → Remeshing, Simplification and Reconstruction → Close Holes**. Se nos desplegará el siguiente panel:

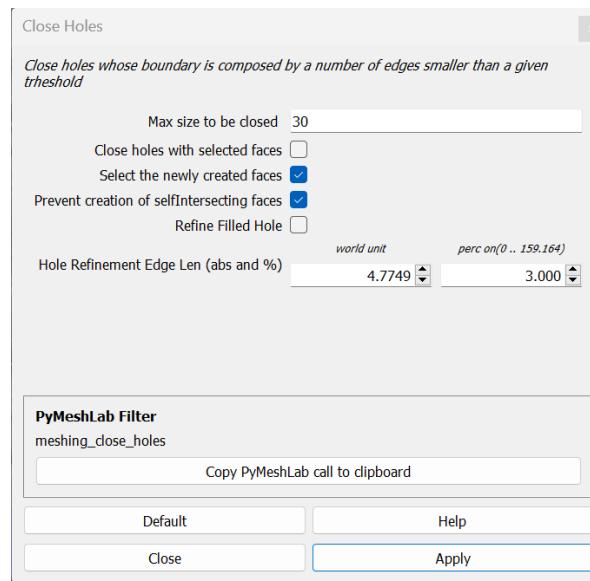


Figura 22: Panel de T-Vértices

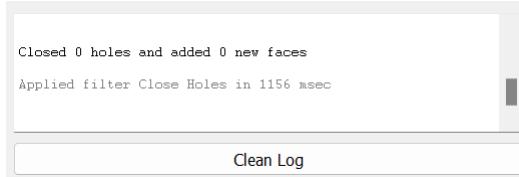


Figura 23: Panel de T-Vértices

Parece que no se ha tapado ningún hueco. Tras esto, estuve probando distintos valores de aristas a tapar, oscilando entre 20 y 70, pero no obtuve otro mensaje en el log, así que parece una malla bastante sólida.

Si se hubieran generado nuevas caras y vértices, habría que repetir todo este proceso de nuevo (*comprobar autointersecciones, eliminar elementos no-variedad, eliminar t-vértices, etc*), para asegurar que la malla queda limpia.

### 3.3. Simplificación

Para agilizar la publicación de nuestro modelo en los medios, es posible que necesitemos reducir su tamaño intentando mantener su aspecto visual, esto puede conseguirse reduciendo el número de vértices y triángulos del modelo.

#### 3.3.1. Simplificación: Quadratic Edge Collapse Decimation

Vamos a elegir este algoritmo por ser el que más respeta los detalles, dentro de lo que hemos visto en el curso. Para ello hacemos click en las siguientes secciones: *Filters → Remeshing, Simplification and Reconstruction → Simplification: Quadratic Edge Collapse Decimation*:

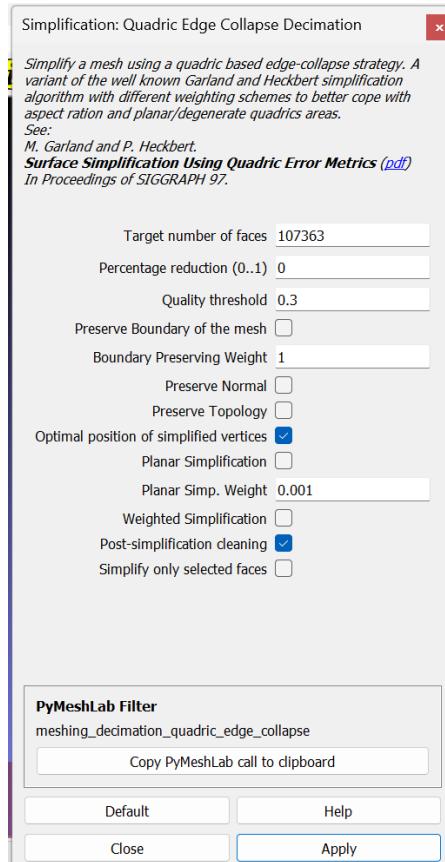


Figura 24: Panel de Simplificación: Quadratic Edge Collapse Decimation

Como es requisito de esta práctica entregar el modelo simplificado al menos un 10 %, actualmente tenemos 1.073.632 caras, por lo que establecemos el valor de “Target Number of Faces” en 107.363 caras. Tras la simplificación, el modelo

tiene 53.683 vértices y 107.362 caras, lo que cumple con la reducción de al menos un 10 %.

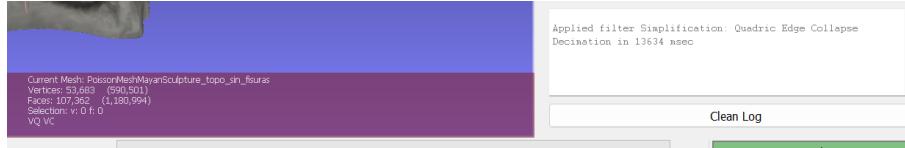


Figura 25: Modelo reducido al 10 %

### 3.3.2. Corrección topológica de la nueva malla

Como tenemos un nuevo modelo, es necesario rehacer una comprobación topológica de los elementos. Como

#### 3.3.2.1 Comprobación de Autointersecciones

Nos aparecen 4 caras, las identifico:

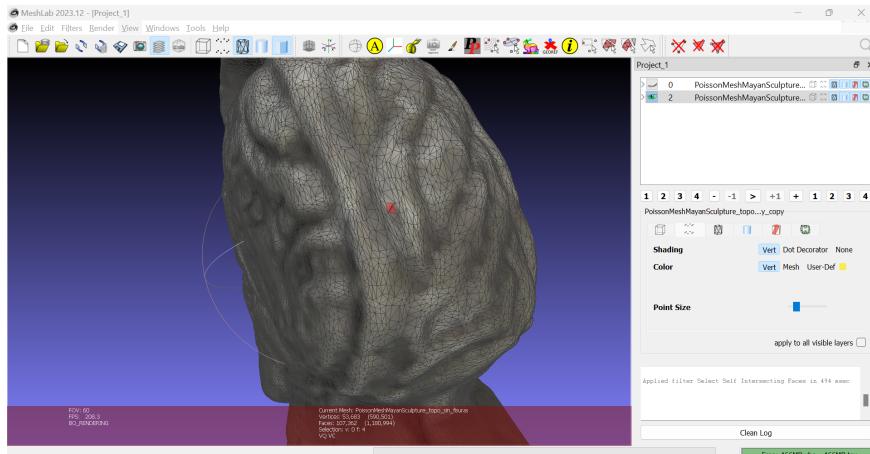


Figura 26: Autointersecciones

Las elimino aprovechando la selección y cierro las fisuras generadas, estableciendo “*Max size to be closed*” a 20. Las deseleccióno y **vuelvo a lanzar el comprobador de autointersecciones**, resultando esta vez en 0. Sigo con el resto de filtros de selección, pero no seleccionan nada:

- Non Mainfold Edges
- Non Mainfold Vertices

Respecto al limpiado y reparación del modelo:

- Remove T-Vertices
- Remove duplicate faces
- Remove duplicate vertices
- Remove isolated pieces (wrt Face Num.)

Durante estas acciones no se han eliminado vértices ni caras, de manera que tenemos el nuevo modelo simplificado topológicamente preparado. Adjunto el detalle de la malla tras la simplificación:

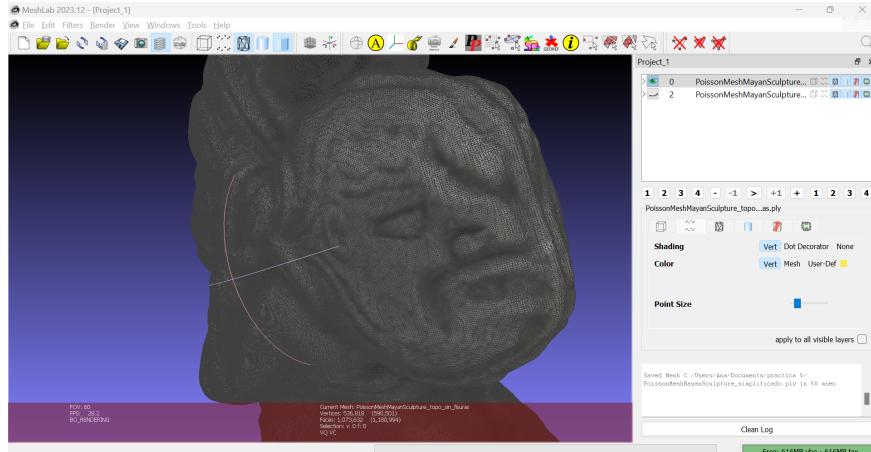


Figura 27: Sin simplificación

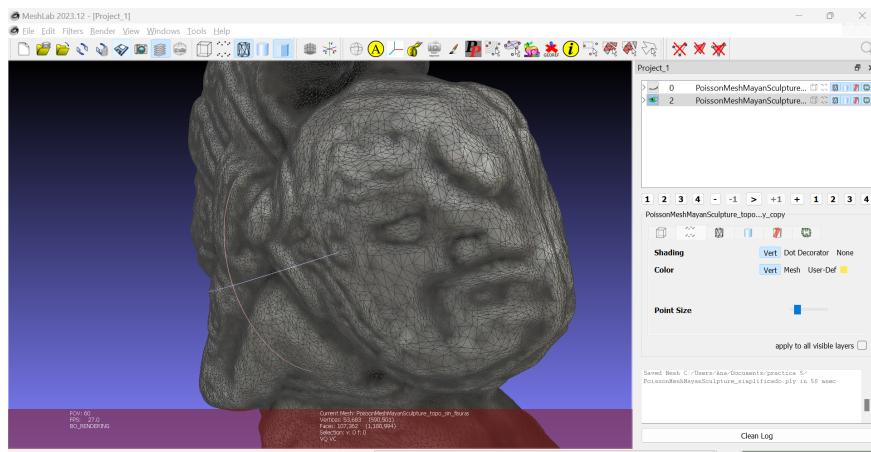


Figura 28: Simplificada

### 3.4. Parametrización

Para realizar el “desenrollado” de la malla y asignarle las coordenadas de textura de la pieza original, vamos a utilizar el algoritmo **Voronoy Atlas**. Para ello hacemos click en las siguientes secciones: **Filters** → **Texture** → **Parametrization: Voronoy Atlas**. Se nos desplegará el siguiente panel, en el que faremos click en “*Apply*”:

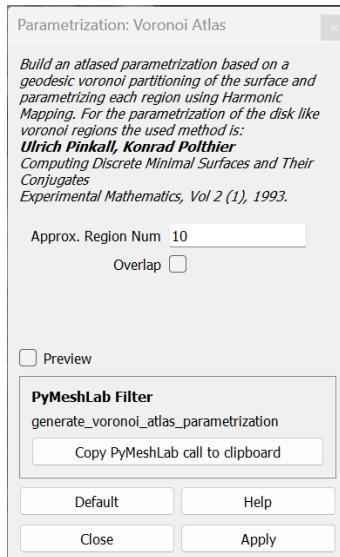


Figura 29: Panel Voronoy Atlas

Tras la ejecución, este es el resultado:

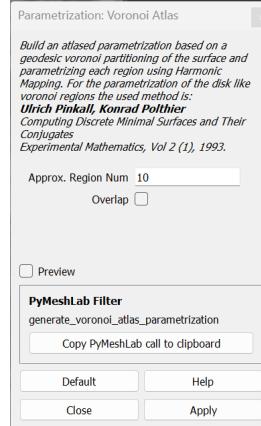


Figura 30: Panel Voronoy Atlas

### 3.5. Texturización del modelo simplificado

Dejo, para la comparación, la malla *VoroAtlas* generada:

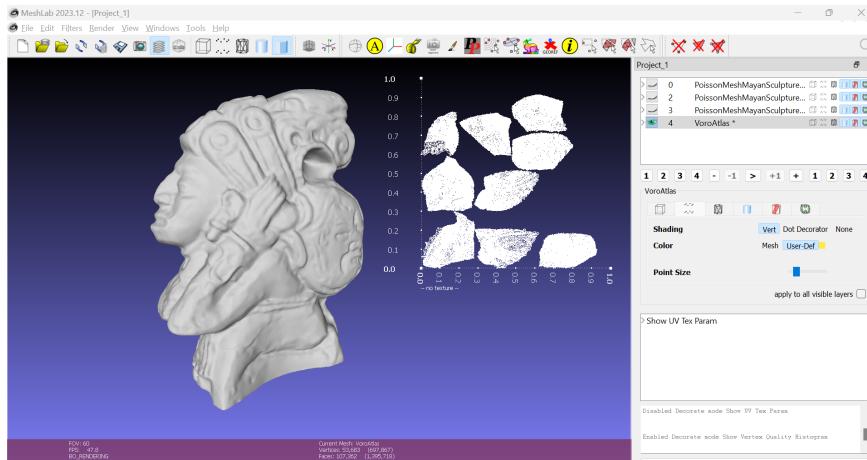


Figura 31: Voronoy Atlas antes de texturar

Para transferir las texturas de la malla original al nuevo modelo simplificado utilizaremos la función, primero importo la malla de nube de puntos (la original), y luego utilizo la función: **Filters → Texture → Transfer: Vertex attributes to texture (1 or 2 meshes)**. Se despliega el siguiente panel:

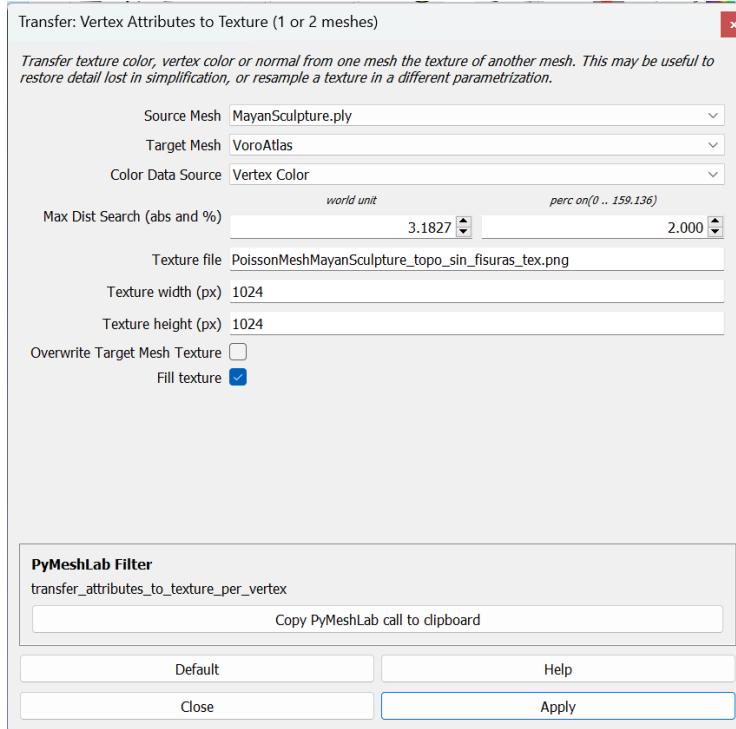


Figura 32: Panel de texturización

Este sería el resultado tras hacer click en “*Apply*” y poner el color de las caras en blanco:

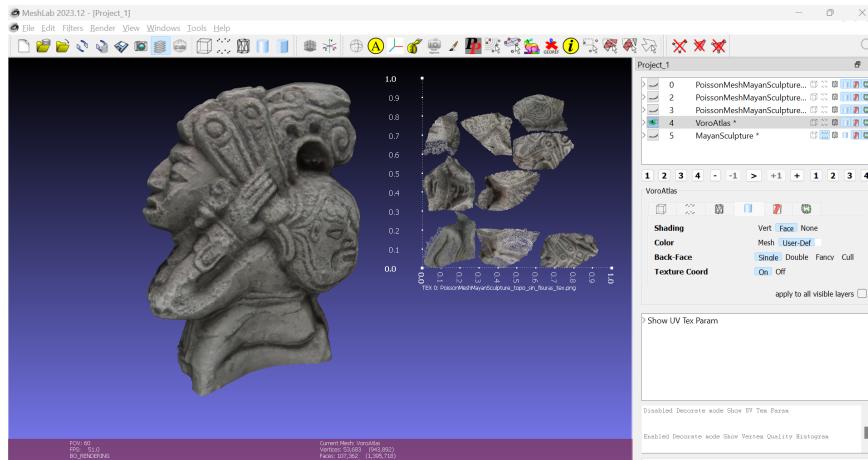


Figura 33: Modelo texturizado

Muestro el resultado de la textura en .png:



Figura 34: Textura

## 3.6. Publicación del modelo

### 3.6.1. Exportar modelo a formato X3D

Para publicar el modelo vamos a necesitar exportar el fichero al formato .x3d, para ello hacemos click en **Files → Export Mesh As...**:

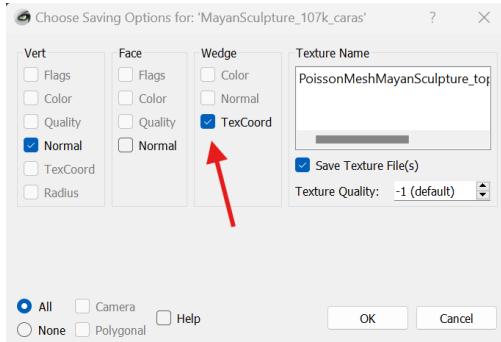


Figura 35: Panel para exportar a .x3d

**Importante:** seleccionar la casilla de exportar las coordenadas de las texturas.

### 3.6.2. Subir nuestro modelo a GitHub

Para publicar nuestro modelo utilizaremos la herramienta de colaboración **GitHub**. Para crear un repositorio y hacer seguimiento de este proyecto haremos click, dentro de nuestro panel principal, en **"New"**:

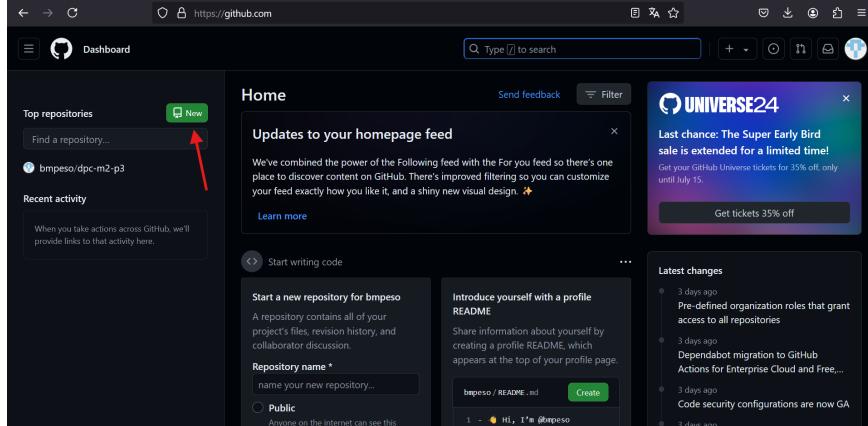


Figura 36: Creación del repositorio 1

A continuación rellenaremos los campos que creamos convenientes (*señalo en recuadros los campos modificados*) y haremos click en **Create repository**:

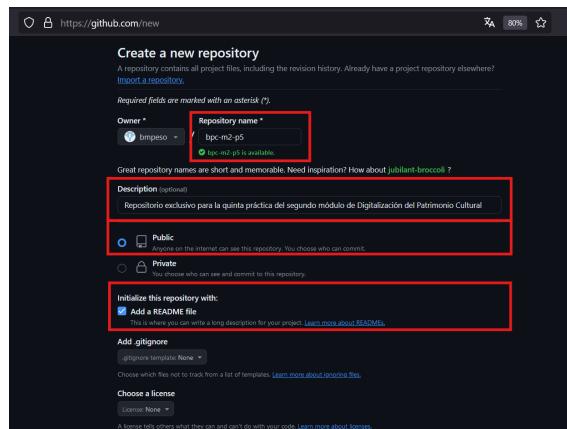


Figura 37: Creación del repositorio 2

### 3.6.3. Publicar nuestro modelo desde GitHub

Para publicar nuestro modelo en GitHub, además de subir los archivos al repositorio y el *index.html*, deberemos configurarlo para que muestre nuestra página web (<https://usuario.github.com>). Para ello iremos a la configuración del repositorio y rellenaremos la siguiente pantalla:

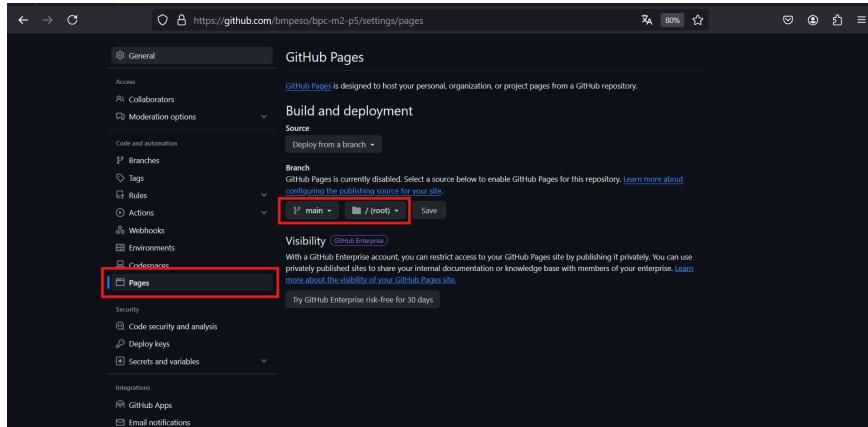


Figura 38: Repositorio → Settings → Pages

Tras esto, podremos navegar a nuestra página:

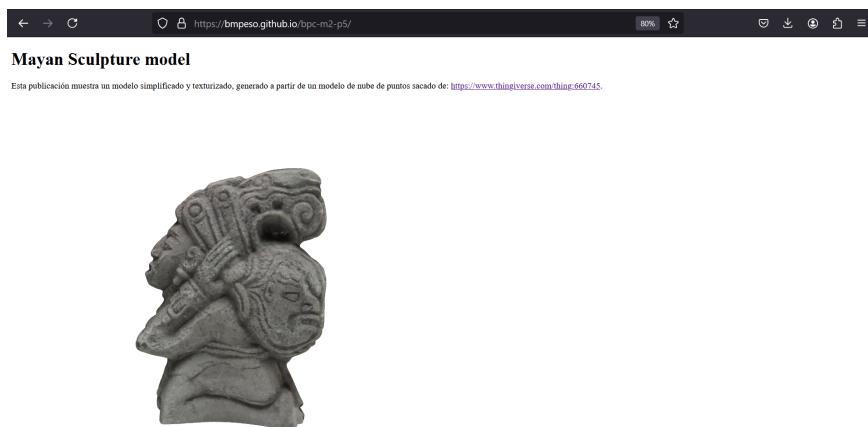


Figura 39: Modelo publicado

## 4. Valoración personal

Hice esta práctica conforme iba viendo los vídeos. Al terminar la práctica he aprendido a valorar varias cosas:

- Cada división de la práctica: esto me ayudará a identificar la situación/el momento en la que me vaya a encontrar cuando obtenga una malla de primeras y quiera publicarla. Ahora tengo una idea general del proceso de publicación de un modelo.
- Concepto de "*simplificación*": Era algo en lo que no había caído y me ha resultado útil aprender a reducir un modelo a un tamaño que yo pueda elegir, manteniendo mejor o peor los detalles en base a necesidades
- Aprender a utilizar GitHub: era consciente de que podían tenerse páginas web, pero no sabía cómo configurarlo

Además, me ha gustado también utilizar una herramienta *open source* como *MeshLab*. De manera que por todo esto y en general diría que el proceso me ha gustado mucho. Dejo los links del repositorio y de la web:

- Repositorio: [bmpeso/bpc-m2-p5](https://github.com/bmpeso/bpc-m2-p5)
- Página del modelo: <https://bmpeso.github.io/bpc-m2-p5>